KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

Solving a time-indexed formulation by preprocessing and cutting planes

Berghman L, Spieksma F, T'Kindt V.

KBI_1407

Solving a time-indexed formulation by preprocessing and cutting planes*

Lotte Berghman† Frits C.R. Spieksma‡
Vincent T'Kindt§

Abstract: We consider a time-indexed formulation for the unrelated parallel machine scheduling problem. We show that all polyhedral knowledge known from the single machine problem (in particular, valid inequalities) is applicable to this formulation. We present new facet-inducing valid inequalities and a preprocessing technique involving fixing variables based on reduced costs. We combine both techniques in a basic cutting-plane algorithm and test the performance of the resulting algorithm by running it on randomly generated instances.

Keywords: unrelated machine scheduling, time-indexed formulation, valid inequalities, cutting plane algorithm, variable fixing.

1 Introduction

The time-indexed formulation for single machine scheduling problems is well studied in the literature. Seminal work of Dyer and Wolsey (1990) and Sousa and Wolsey (1992), and further work by Crama and Spieksma (1996) and van den Akker et al. (1999) have resulted in a large body of polyhedral results for the time-indexed formulation. Generally speaking, the major advantage of a time-indexed formulation is the tight LP-bound, while the greatest disadvantage are the large number of variables, especially when processing times are large. One possible avenue to, at least partially, overcome this difficulty is using column generation, as was done in van den Akker et al. (2000) and Bigras et al. (2008). An arc-time indexed formulation is an extended formulation that yields strictly better bounds than the time-indexed formulation at the cost of an even larger number of variables, one for each pair of jobs and each possible completion time (see, e.g. Sourd 2009; Tanaka et al. 2009).

As far as we are aware, all this polyhedral knowledge has not been applied to time-indexed formulations of scheduling problems with multiple machines, in particular unrelated parallel machine scheduling. This is confirmed by Unlu and Mason (2010) who evaluate integer programming formulations for parallel machine scheduling and recommend to use

*This work is supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office, and by FWO grant G.0729.13.

†Université de Toulouse - Toulouse Business School, 20 BD Lascrosses BP 7010, 31068 Toulouse Cedex 7, France. Email: l.berghman@tbs-education.fr

‡ORSTAT, Faculty of Economics and Business, KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium.

§Université François-Rabelais de Tours, Laboratoire d'Informatique, 64 avenue Jean Portalis, 37200 Tours, France.

a time-indexed formulation when job processing times are small. Moreover, they explicitly suggest to develop valid inequalities.

This paper deals with the time-indexed formulation of the unrelated parallel machine scheduling problem, where the processing cost of a job is an arbitrary function of its starting time. Notice that this allows to model many objective functions such as (weighted) sum of completion times or makespan, and to incorporate features such as release times and precedence relations. Our goal is (1) to point out that all polyhedral knowledge existing for single-machine problems can be applied to multi-machine problems, (2) to describe a new class of facet-inducing inequalities for the time-indexed formulation for multiple machines, (3) to implement a preprocessing technique that uses variable fixing based on reduced costs, and (4) to show the computational performance of an algorithm that combines valid inequalities and variable fixing by testing this algorithm on randomly generated instances.

The problem statement and the proposed single machine scheduling formulation are presented in Section 2. We show in Section 3 that existing valid inequalities can be applied to our formulation. In Section 4, we present a new class of facet-inducing valid inequalities and Section 5 describes the preprocessing technique based on variable fixing. Section 6 presents our final algorithms and the outcome of running them on randomly generated instances, while Section 7 contains the conclusions.

2 Integer programming formulations

Consider the problem of scheduling n jobs on a single machine within a given timespan. The timespan $[0, T]$ is discretized into T time periods of length one. Period t refers to the time slot $[t-1, t]$; $t = 1, \ldots, T$. The processing time of job j equals p_j. The machine can handle at most one job at a time and preemption is not allowed. When job j starts in time period t, a known cost of c_{jt} is incurred. The problem is to find a schedule that minimizes total cost.

This problem can be modeled as follows: for each job j and for each time period $t = 1, \ldots, T$, we define

$$x_{jt} = \begin{cases} 1 & \text{if the processing of job } j \text{ starts in time period } t, \\ 0 & \text{otherwise.} \end{cases}$$

The well-known time-indexed formulation for the single machine scheduling problem (as presented in Sousa and Wolsey (1992) and van den Akker et al. (1999)) is the following:

$$\min \sum_{j=1}^{n} \sum_{t=1}^{T} c_{jt} x_{jt} \tag{1}$$

subject to

$$\sum_{t=1}^{T} x_{jt} = 1 \qquad \forall j = 1, \ldots, n, \tag{2}$$

$$\sum_{j=1}^{n} \sum_{s=t-p_j+1}^{t} x_{js} \leq 1 \qquad \forall t = 1, \ldots, T, \tag{3}$$

$$x_{jt} \in \{0, 1\} \qquad \forall j = 1, \ldots, n; \forall t = 1, \ldots, T. \tag{4}$$

The objective function (1) minimizes the total cost. Constraints (2) state that each job has to be scheduled exactly once and constraints (3) express that during each time period t, only one job can be executed; we refer to (3) as the *capacity constraints*. This formulation is often called pseudo-polynomial because the number of variables and the number of constraints depend on the length of the time horizon. Thus, indeed if processing times are large, the number of variables grows. However, notice that (1) the problem is already strongly NP-hard if $p_j = 2$ for all j (Crama and Spieksma 1996) and (2) there exist applications where the cost of starting a job is 'truly' arbitrary, see e.g. the assignment of feeders to a component placement machine (Crama et al. 1990) or the assignment of ships to berths in container terminals (Hansen et al. 2008), leading to an input of $O(nT)$ numbers.

When one wants to generalize this formulation to the identical parallel machine scheduling problem, the right-hand side of constraints (3) can be set to m, the number of machines. However, when the machines are not identical, i.e., when a job's processing time depends on the machines, such a trick is not longer possible.

We now consider the problem of scheduling n jobs on m unrelated parallel machines within a given timespan. Again, each machine can handle at most one job at a time and preemption is not allowed. The processing time of a job now depends on the machine: the processing time of job i on machine k is denoted by p_{ik}. The processing cost of a job depends both on the machine and the time period in which the job is started: the processing cost of job i when executed at machine k and started at time period t is denoted by c_{ikt}. Again, we are interested in a feasible schedule minimizing total cost.

Unrelated parallel machine scheduling has received quite some attention in literature, especially the special case where one wants to minimize total weighted completion time. We will not review this literature, we simply mention Lenstra et al. (1990) and Gairing et al. (2007) and the references contained in those papers. Also, Vredeveld and Hurkens (2002) present an empirical comparison of different polynomial-time approximation algorithms and local search heuristics for the problem of minimizing total weighted completion time on unrelated parallel machines. The algorithms are based on rounding a fractional solution to an LP-relaxation or to a convex quadratic-programming relaxation.

We will model this unrelated parallel machine scheduling problem as a single machine problem in the following way: by copying each job m times, we obtain nm tasks j. We define J as the set containing all tasks. This set can be partitioned in two different ways. First of all, we consider the subsets $J_i \subseteq J$ with $i = 1, \ldots, n$ containing all tasks related to job i. Secondly, we consider the subsets $J^k \subseteq J$ with $k = 1, \ldots, m$ containing all tasks related to machine k. Every subset $J_i \cap J^k$ consists of a single task j. The processing time of task $j = J_i \cap J^k$ equals $p_j = p_{ik}$. We denote by $c_{jt} = c_{ikt}$ the cost of starting task $j = J_i \cap J^k$ in

time period t. Notice that specifying the task (index j), implies specifying the job and the machine, and vice versa.

For each task j and for each time period $t = 1, \ldots, T - p_j + 1$, we define the decision variables

$$x_{jt} = \begin{cases} 1 & \text{if task } j \text{ starts in time period } t, \\ 0 & \text{otherwise.} \end{cases}$$

An IP-model for this machine scheduling problem is the following:

$$\min \sum_{j=1}^{nm} \sum_{t=1}^{T-p_j+1} c_{jt} x_{jt} \tag{5}$$

subject to

$$\sum_{j \in J_i} \sum_{t=1}^{T-p_j+1} x_{jt} = 1 \qquad \forall i = 1, \ldots, n, \tag{6}$$

$$\sum_{j \in J^k} \sum_{s=t-p_j+1}^{t} x_{js} \leq 1 \qquad \forall k = 1, \ldots, m; \forall t = \min_{j \in J_k} p_j, \ldots, T, \tag{7}$$

$$x_{jt} \in \{0, 1\} \qquad \forall j = 1, \ldots, nm; \forall t = 1, \ldots, T - p_j + 1. \tag{8}$$

The objective function (5) minimizes the total cost. Constraints (6) state that out of the tasks related to job i, i.e., J_i, exactly one task has to be scheduled. The capacity constraints are formulated using constraints (7): for each time period t, only one task out of the tasks related to machine k, i.e., J^k, can be executed. We obtain the LP-relaxation of this formulation by replacing constraints (8) by the following one: $\forall j = 1, \ldots, nm; \forall t = 1, \ldots, T - p_j + 1 : 0 \leq x_{jt} \leq 1$.

Notice that, in the case of a single machine, i.e. when $m = 1$, formulation (5)-(8) becomes (1)-(4). In the remainder of this text, we will use P_m to denote the convex hull of feasible solutions of (6)-(8).

3 Known valid inequalities

In this section, we review the known valid inequalities for P_1. Notice that an inequality for P_1 can be extended to an inequality for P_m ($m > 1$) by setting all coefficients that correspond to variables that involve tasks not related to some specific machine k ($1 \leq k \leq m$) to 0. Then, it is not difficult to observe that in this way, any inequality valid for P_1 can be extended to an inequality valid for P_m. We record this observation formally:

Fact 1 *Any inequality valid for P_r is valid for P_m, for each $r \leq m$.*

Proof: We argue by contradiction. Suppose there is an inequality valid for P_r which - when extended - is not valid for P_m. Hence, a feasible solution to the m-machine problem is cut off by the valid inequality. However, a feasible solution to an instance of the m-machine problem, when restricted to a subset of r machines, becomes a feasible solution to an instance of the

r-machine problem. Thus, we have identified a feasible solution to the r-machine problem that is cut off by the extended inequality, and hence also by the original inequality. This contradicts the initial assumption. □

Fact 1 motivates us to formulate the known inequalities in terms of P_m. To do so, we need the following notation. For each $j = 1, \ldots, nm$, we define $T(j)$ as the set of tasks that are related to the same machine as task j. Notice that $T(j)$ does not include task j. Moreover, we define $p_j^* = \max_{l \in T(j)} p_l$; thus p_j^* is the largest processing time of the tasks in $T(j)$.

Sousa and Wolsey (1992) give the following inequalities. For each time period $t = 1, \ldots, T$, for each task $j = 1, \ldots, nm$ and for each $\Delta \in \{2, \ldots, p_j^*\}$:

$$\sum_{s=t-p_j+1}^{t+\Delta-1} x_{js} + \sum_{l \in T(j)} \sum_{s=t-p_l+\Delta}^{t} x_{ls} \leq 1 \tag{9}$$

In inequality (9), task j is sometimes called the 'special' task. These inequalities are known to be facet-defining for P_1 (Sousa and Wolsey 1992), and in fact they constitute all facet-defining inequalities for P_1 with integral coefficients and right-hand side 1, (see van den Akker et al. 1999).

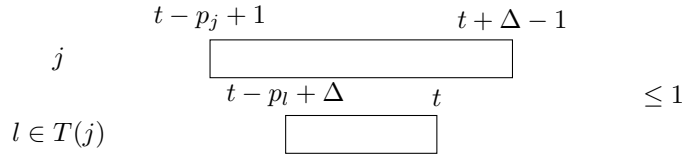
To give a pictorial description of this inequality, we will use a similar notation as van den Akker et al. (1999). The index-set of variables with nonzero coefficients in an inequality is denoted by V. The set of nonzero coefficients in an inequality associated with task j defines a set of time periods $V_j = \{t | (j,t) \in V\}$. Thus the union over all j of all V_j equals V. We define an interval $[a,b]$ as the set of periods $\{a, a+1, \ldots, b\}$. If $a > b$, then $[a,b] = \emptyset$. We shall represent inequalities by diagrams. A diagram contains a line for each task. The blocks on the line associated with task j indicate the time periods t for which x_{jt} occurs in the inequality.

Inequalities (9) of Sousa and Wolsey (1992) use the following time periods:

for task j: $V_j = [t - p_j + 1, t + \Delta - 1]$,
for each task $l \in T(j)$: $V_l = [t - p_l + \Delta, t]$,

where $\Delta \in \{2, \ldots, p_j^*\}$.

These inequalities can be represented by the following diagram.



Using this diagram, it is relatively easy to see that inequalities (9) are valid. Indeed, notice that if some task $l \in T(j)$ starts at some time in V_l, no other task from $T(j)$ can start in V_l (since both tasks would be active at time t). Also task j cannot start in V_j, since starting task j directly after the completion of task l is impossible: task l is active until $t + \Delta - 1$; starting task j before the beginning of task l is equally impossible, since even

starting task j at $t - p_j + 1$ means that task j is active at time t. This implies validity of (9).

A lot more is known concerning the facial structure of P_1. Sousa and Wolsey (1992) already give other classes of facet-defining inequalities, Crama and Spieksma (1996) find classes of facet-defining inequalities that apply to the case of equal processing times, and van den Akker et al. (1999) present three classes of facet-defining inequalities that collectively constitute all facet-defining inequalities with integral coefficients that have right-hand side 2. We will not give an explicit description of these inequalities, however, let us emphasize here that any facet-defining inequalities for P_1 other than an inequality from Sousa and Wolsey (1992) has right-hand side 2 or more. Moreover observe that all these inequalities deal with a single machine. In the next section, we exhibit a class of valid inequalities that specifically focus on the presence of multiple machines. Indeed, this is the first description of a class of valid inequalities that contains variables corresponding to different machines.

4 A new class of valid inequalities

In this section, we introduce a new class of valid inequalities that contains variables corresponding to different machines.

4.1 Example

We first specify an instance. Let $n = 3$, $m = 2$, $T = 14$, $J^1 = \{1, 2, 3\}$, $J^2 = \{4, 5, 6\}$, $J_1 = \{1, 4\}$, $J_2 = \{2, 5\}$, $J_3 = \{3, 6\}$, $p_1 = 4$, $p_2 = 3$, $p_3 = 5$, $p_4 = 1$, $p_5 = 5$ and $p_6 = 2$, Further, the c_{jt} coefficients are given in Table 1 where a row corresponds to a task, and a column corresponds to a time period.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	2	2	2	1	2	2	2	2	2	2	2	2	2
2	1	1	1	1	1	1	1	0	1	1	1	1	1
3	1	1	1	1	1	1	0	1	1	1	1	1	1
4	2	2	2	2	1	2	2	2	2	2	2	2	2
5	0	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 1: The coefficients c_{jt} for the example instance.

When solving the LP-relaxation (5) - (8) of this instance, we find the fractional solution $x_{1,4} = x_{2,8} = x_{3,7} = x_{4,5} = x_{5,1} = x_{6,2} = \frac{1}{2}$. We claim that this solution is not cut off by any known facet-defining inequality. Indeed, observe that the sum of the variables corresponding to jobs on machine 1, i.e., the variables corresponding to jobs $1, 2, 3$, sum up to $\frac{3}{2}$. Hence, this partial solution cannot be eliminated by any facet-defining inequality other than an inequality from (9), since all known facet-defining inequalities other than (9) have right-hand side 2 or more. In addition, we leave it to the reader to verify that inequalities (9) also

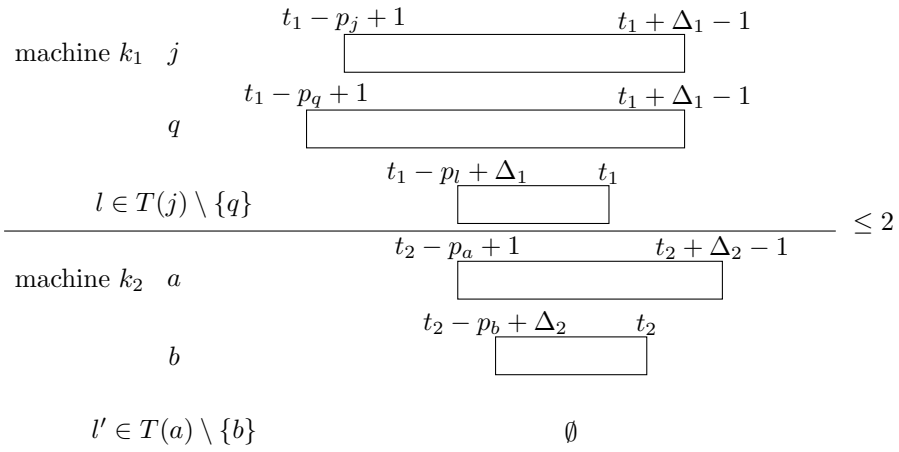


Figure 1: The diagram representing the valid inequalities (10).

do not cut away this particular solution. A similar argument holds for the jobs corresponding to machine 2.

4.2 A new class of valid inequalities

For each pair of jobs $\{i_1, i_2\} \in \{1, \ldots, n\}$, and for each pair of machines $\{k_1, k_2\} \in \{1, \ldots, m\}$, let $j = J_{i_1} \cap J^{k_1}$, $q = J_{i_2} \cap J^{k_1}$, $a = J_{i_1} \cap J^{k_2}$ and $b = J_{i_2} \cap J^{k_2}$. For each quadruple of such four tasks, for all time periods $t_1, t_2 = 1, \ldots, T$, for all $\Delta_1 \in \{2, \ldots, \min\{p_j^*, p_q^*\}\}$ and for all $\Delta_2 \in \{2, \ldots, p_a^*\}$, we have the following inequalities:

$$\sum_{s=t_1-p_j+1}^{t_1+\Delta_1-1} x_{js} + \sum_{s=t_1-p_q+1}^{t_1+\Delta_1-1} x_{qs} + \sum_{l \in T(j)\setminus\{q\}} \sum_{s=t_1-p_l+\Delta_1}^{t_1} x_{ls} + \sum_{s=t_2-p_a+1}^{t_2+\Delta_2-1} x_{as} + \sum_{s=t_2-p_b+\Delta_2}^{t_2} x_{bs} \leq 2 \tag{10}$$

These inequalities can be represented by the diagram presented in Figure 1.

Theorem 2 *Inequalities (10) are valid inequalities for P_m, for each $m \geq 2$.*

Proof: Take two equalities of type (6), one for job i_1, and one for job i_2. Take two inequalities of type (7), one for machine k_1 and time period t_1, and one for machine k_1 and time period $t_1 + \Delta_1 - 1$ (with $\Delta_1 \in \{2, \ldots, \min\{p_j^*, p_q^*\}\}$). Take an inequality of type (9) for machine k_2, period t_2, job i_1 and $\Delta_2 \in \{2, \ldots, p_a^*\}$. Assume that we have a multiplier equal to $\frac{1}{2}$ for each of these five inequalities. If we apply integer rounding on both sides of the resulting inequality, we obtain the inequality (10). □

Notice that the Chvatal rank of these inequalities does not exceed 2; further, there are $\mathcal{O}(n^2 m^2 T^2 p_{\max}^2)$ inequalities in the new class. The inequalities cannot be strengthened, as witnessed by our next result:

Theorem 3 *Inequalities (10) are facet-defining inequalities for P_m, for each $m \geq 2$.*

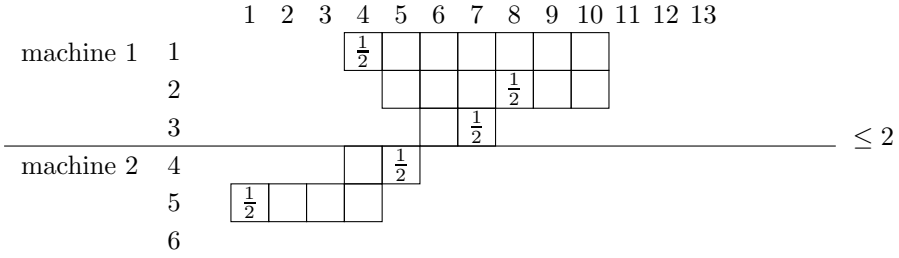
We give a proof of this result in the Appendix.

4.3 Example continued

With $j = 1$, $q = 2$, $a = 4$, $b = 5$, $t_1 = 7$, $\Delta_1 = 4$, $t_2 = 4$ and $\Delta_2 = 2$, inequality (10) boils down to

$$x_{1,4} + x_{1,5} + x_{1,6} + x_{1,7} + x_{1,8} + x_{1,9} + x_{1,10} + x_{2,5} + x_{2,6} + x_{2,7} + x_{2,8} \\ + x_{2,9} + x_{2,10} + x_{3,6} + x_{3,7} + x_{4,4} + x_{4,5} + x_{5,1} + x_{5,2} + x_{5,3} + x_{5,4} \leq 2.$$

It is displayed by the squared blocks in the figure below, and cuts off the fractional solution.



5 Preprocessing the IP formulation

Preprocessing is a general process which aims to solve an IP-formulation more efficiently by using structural properties of the problem. In this section, we show how a well-known preprocessing technique, namely variable fixing, can be applied to our time indexed formulation. Variable fixing is a technique that is able to reduce the number of $0-1$ variables in Formulation (5)-(8), and it is expected that the reduced IP formulation will be solved faster than the initial formulation. In addition, variable fixing allows to find stronger LP-bounds. Variable fixing is a technique based on simple links between an IP formulation and its linear relaxation, and goes back to Tomlin (1971). More recent applications of this technique can be found in Baptiste et al. (2010) and T'kindt et al. (2007).

We write x^{LP} for the values of the x-variables of the LP-relaxation of (5)-(8), and we write x^{IP} for the values of the x-variables of the IP-formulation (5)-(8). Let z^*_{LP} (resp z^*_{IP}) be the value of the corresponding optima and let B^* be the associated basis. When fixing our variables x^{IP}_{jt} we distinguish between the basic variables and the non-basic variables.
Let us first consider non-basic variables $x^{IP}_{jt} \notin B^*$. It is well known that, applied to our model, we have:

$$z^*_{MIP} = z^*_{LP} + \sum_{x^{IP}_{jt} \notin B^*} r_{jt} x^{IP}_{jt}, \tag{11}$$

with r_{jt} the reduced cost associated to variable x^{IP}_{jt} (see Tomlin (1971)). Let UB be an upper bound to z^*_{MIP}. Then, we can write:

$$\sum_{x^{IP}_{jt} \notin B^*} r_{jt} x^{IP}_{jt} \leq UB - z^*_{LP}. \tag{12}$$

From inequality (12) we can deduce the following fixing rule: $\forall x_{jt}^{IP} \notin B^*$, if $r_{jt} > UB - z_{LP}^*$ then in any optimal solution of the IP formulation, $x_{jt}^{IP} = 0$. By reasoning on the slack variables s_{jt} associated to the constraints $x_{jt} \leq 1$, we can deduce when $x_{jt}^{IP} = 1$: if a non-basic slack variable s_{jt} has to be fixed to 0, then the associated variable x_{jt}^{IP} is fixed to 1.

For basic variables $x_{jt}^{IP} \in B^*$ all reduced costs are equal to 0, which makes the above fixing procedure inefficient. Therefore, we use pseudo-costs l_{jt} and u_{jt} computed, for instance, by means of Driebeek's penalties (see Tomlin (1971) for more details). These pseudo-costs are computed for basic variables starting from the reduced costs of the non-basic variables. The meaning of these penalties is the following: l_{jt} (resp. u_{jt}) is a unitary lower estimate on the increase of z_{LP}^* if x_{jt}^{IP} is set to 0 (resp. 1), such that $\forall x_{jt}^{IP} \in B^* : z_{LP}^* + l_{jt} x_{jt}^{LP} \leq UB$ and $\forall x_{jt}^{IP} \in B^* : z_{LP}^* + u_{jt}(1 - x_{jt}^{LP}) \leq UB$. We have:

1. $\forall x_{jt}^{IP} \in B^*$, if $(l_{jt} x_{jt}^{LP}) > (UB - z_{LP}^*)$ then $x_{jt}^{IP} = 1$,
2. $\forall x_{jt}^{IP} \in B^*$, if $(u_{jt}(1 - x_{jt}^{LP})) > (UB - z_{LP}^*)$ then $x_{jt}^{IP} = 0$.

The efficiency of these two variable fixing techniques is strongly influenced by the gap between UB and z_{LP}^*, and the value of the reduced costs. To strengthen these techniques we can use valid inequalities like the ones presented in section 4. The choice of the included valid inequalities and of the upper bound UB are reported in section 6.

The preprocessing algorithm works as described in Algorithm 1.

Algorithm 1 Preprocessing algorithm; inputs: *valid inequalities* and UB

Add the *valid inequalities* to the LP relaxation.
Solve the LP relaxation: let z_{LP}^0 be the optimal solution value.
Fix basic and non-basic variables with UB.
Solve the LP relaxation: let z_{LP}^1 be the optimal solution value.
while $(z_{LP}^0 < z_{LP}^1)$ **do**
  Fix basic and non-basic variables with UB.
  $z_{LP}^0 = z_{LP}^1$.
  Solve the LP relaxation: let z_{LP}^1 be the optimal solution value.
end while
return z_{LP}^1 and the associated variable values x_{LP}^1

Remark that we can reduce the m-machine problem to a single machine problem by concatenating the timespan of the m machines to obtain a large timespan spanning mT periods. The processing times now become dependent upon the particular period: for each of the first T periods, the processing time of job j equals p_{j1}, then p_{j2} for the next T periods, and so on. All results of this paper are also valid for the single machine case with arbitrary period-dependent processing times as the presented m-machine problem is a special case of this single machine problem.

6 Computational evaluations

A series of computational evaluations have conducted in order to evaluate the impacts of the valid inequalities and the preprocessing algorithm on: (1) the linear relaxation LP of the IP formulation; (2) the solution of the IP formulation. We first provide details about the generation of our instances (section 6.1), before discussing the obtained results on the LP (section 6.3) and on the IP (6.4).

6.1 Generation of the instances

The IP formulation of instances that are generated according to Sousa and Wolsey (1992) and van den Akker et al. (1999) are almost always easy to solve by CPLEX, especially as the number of machines increase. For this reason, we introduce another generation scheme which leads to harder instances for the parallel machine problem. The idea of such a scheme is to increase the number of resource conflicts when trying to minimize the objective function.

The number of jobs n is taken in the set $\{100, 150, 200\}$ and the number of machines m in the set $\{1, 2, 3, 5, 10\}$. The time horizon is defined as $|T| = 1.25 \times \frac{p_{\max}+1}{2} \times \frac{n}{m}$, with $p_{max} = 20$ the maximal processing time value. The processing times on machines are uniformly distributed in $[1, p_{\max}]$. The hardness of the instances comes from the generation of the processing costs c_{jt}. The scheduling horizon $[0; T]$ is splitted into $\sqrt{n}$ equal-size intervals $[T_i; T_{i+1}]$ and $\sqrt{n}$ jobs take their minimum c_{jt} values in each interval. The size of each interval, denoted by $SizeInt$, is equal to $\lceil \frac{T}{\sqrt{n}} \rceil$ except the last one which can be slightly smaller due to the rounding. Then, c_{jt} are drawn at random in the interval $[0; 10 * T - G(\mu_{jt}, \sigma)]$ with $G(\mu, \sigma)$ referring to a normal distribution of mean μ_{jt} and variance σ. We experimentally set $\mu_{jt} = \frac{9*T}{|t\%SizeInt - j\%\sqrt{n}|+1}$ and $\sigma = 1.3 * SizeInt$. An illustration of the distribution of the processing costs c_{jt} is given in figure 2.
For each combination of n and m, 20 instances were created, yielding 300 instances in total.

6.1.1 Testing environment

All algorithms are encoded in C using the Microsoft Visual Studio programming environment, and executed on a PC computer with an Intel Core i5 CPU 4 Core 2.6-GHz processor and 8 GB RAM, equipped with Windows 7. CPLEX version 12.2 is used to solve the IP and LP models, and is configured to use only 1 thread. Besides, when solving the IP model, a time limit of 3600 seconds and a memory limit of 1Gb is imposed: whenever one of these limits is reached before the end of the solution, then the solver is assumed to have failed to solve the corresponding instance.

6.2 Separation

The separation algorithm for inequalities (10) basically enumerates all possible valid inequalities and tests whether they are violated. Its operation is similar to the separation algorithm described in van den Akker et al. (1999) and uses the necessary conditions of Fact 4. Notice that the vector $\tilde{x}$, the current LP-solution, will be sparse; this fact is used in the separation.

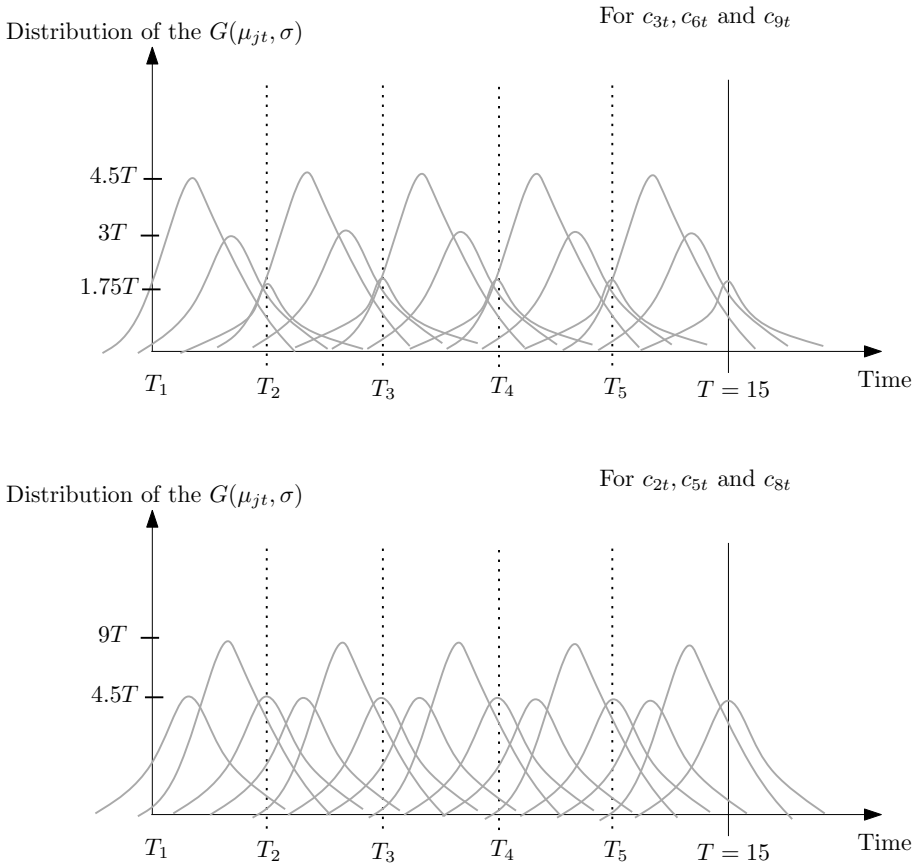


Figure 2: Example of the distribution of the processing costs with $n = 9$ and $T = 15$

For instance, we argue hereunder that, in order for an inequality of type (10) to be violated, each of the five terms that together form inequality (10) must have a value strictly between 0 and 1.

Fact 4 *Some necessary conditions for having a violated inequality of type (10):*
Condition 1: $0 < \sum_{s=t_1-p_j+1}^{t_1+\Delta_1-1} x_{js} < 1$
Condition 2: $0 < \sum_{s=t_1-p_q+1}^{t_1+\Delta_1-1} x_{qs} < 1$
Condition 3: $0 < \sum_{l \in T(j)\setminus\{q\}} \sum_{s=t_1-p_l+\Delta_1}^{t_1} x_{ls} < 1$
Condition 4: $0 < \sum_{s=t_2-p_a+1}^{t_2+\Delta_2-1} x_{as} < 1$
Condition 5: $0 < \sum_{s=t_2-p_b+\Delta_2}^{t_2} x_{bs} < 1$
Condition 6: $0 < \sum_{s=t_1-p_j+1}^{t_1-p_j+\Delta_1-1} x_{js} + \sum_{s=t_1-p_q+1}^{t_1-p_q+\Delta_1-1} x_{qs} < 1$
Condition 7: $0 < \sum_{s=t_1+1}^{t_1+\Delta_1-1} (x_{js} + x_{qs}) < 1$

Proof: Remark that
(i) $\sum_{s=t_1-p_j+\Delta}^{t_1+\Delta_1-1} x_{js} + \sum_{s=t_1-p_q+\Delta}^{t_1+\Delta_1-1} x_{qs} + \sum_{l \in T(j)\setminus\{q\}} \sum_{s=t_1-p_l+\Delta_1}^{t_1} x_{ls} \leq 1$ because of the capacity of time period $t_1 + \Delta_1 - 1$
(ii) $\sum_{s=t_1-p_j+1}^{t_1} x_{js} + \sum_{s=t_1-p_q+1}^{t_1} x_{qs} + \sum_{l \in T(j)\setminus\{q\}} \sum_{s=t_1-p_l+\Delta_1}^{t_1} x_{ls} \leq 1$ because of the capacity of time period t_1
(iii) $\sum_{s=t_2-p_a+1}^{t_2+\Delta_2-1} x_{as} + \sum_{s=t_2-p_b+\Delta_2}^{t_2} x_{bs} \leq 1$ as it is an equality of type (9).
(iv) $\sum_{s=t_1-p_q+1}^{t_1+\Delta_1-1} x_{qs} + \sum_{l \in T(j)\setminus\{q\}} \sum_{s=t_1-p_l+\Delta_1}^{t_1} x_{ls} \leq 1$ as it is an equality of type (9).
(v) $\sum_{s=t_1-p_j+1}^{t_1+\Delta_1-1} x_{js} + \sum_{l \in T(j)\setminus\{q\}} \sum_{s=t_1-p_l+\Delta_1}^{t_1} x_{ls} \leq 1$ as it is an equality of type (9).
(vi) $\sum_{s=t_1-p_q+1}^{t_1+\Delta_1-1} x_{qs} + \sum_{s=t_2-p_b+\Delta_2}^{t_2} x_{bs} \leq 1$ as we will schedule this job only once.
(vii) $\sum_{s=t_1-p_j+1}^{t_1+\Delta_1-1} x_{js} + \sum_{s=t_2-p_a+1}^{t_2+\Delta_2-1} x_{as} \leq 1$ as we will schedule this job only once.
We argue by contradiction. We show that if one of the necessary conditions is not fulfilled, than (10) cannot be violated.
Condition 1: If $\sum_{s=t_1-p_j+1}^{t_1+\Delta_1-1} x_{js} = 0$ then the solution satisfies (10) because of (iii) and (iv). If $\sum_{s=t_1-p_j+1}^{t_1+\Delta_1-1} x_{js} = 1$ then $\sum_{l \in T(j)\setminus\{q\}} \sum_{s=t_1-p_l+\Delta_1}^{t_1} x_{ls} = 0$ because of (v) and $\sum_{s=t_2-p_a+1}^{t_2+\Delta_2-1} x_{as} = 0$ because of (vii) so the solution satisfies (10) because of (vi).
Condition 2: If $\sum_{s=t_1-p_q+1}^{t_1+\Delta_1-1} x_{qs} = 0$ then the solution satisfies (10) because of (iii) and (v). If $\sum_{s=t_1-p_q+1}^{t_1+\Delta_1-1} x_{qs} = 1$ then $\sum_{l \in T(j)\setminus\{q\}} \sum_{s=t_1-p_l+\Delta_1}^{t_1} x_{ls} = 0$ because of (iv) and $\sum_{s=t_2-p_b+\Delta_2}^{t_2} x_{bs} = 0$ because of (vi) so the solution satisfies (10) because of (vii).
Condition 3: If $\sum_{l \in T(j)\setminus\{q\}} \sum_{s=t_1-p_l+\Delta_1}^{t_1} x_{ls} = 0$ then the solution satisfies (10) because of (vi) and (vii). If $\sum_{l \in T(j)\setminus\{q\}} \sum_{s=t_1-p_l+\Delta_1}^{t_1} x_{ls} \geq 1$ then $\sum_{s=t_1-p_q+1}^{t_1+\Delta_1-1} x_{qs} = 0$ because of (iv) and $\sum_{s=t_1-p_j+1}^{t_1+\Delta_1-1} x_{js} = 0$ because of (v) so the solution satisfies (10) because of (iii).
Condition 4: If $\sum_{s=t_2-p_a+1}^{t_2+\Delta_2-1} x_{as} = 0$ then the solution satisfies (10) because of (v) and (vi). If $\sum_{s=t_2-p_a+1}^{t_2+\Delta_2-1} x_{as} = 1$ then $\sum_{s=t_2-p_b+\Delta_2}^{t_2} x_{bs} = 0$ because of (iii) and $\sum_{s=t_1-p_j+1}^{t_1+\Delta_1-1} x_{js} = 0$ because of (vii) so the solution satisfies (10) because of (iv).
Condition 5: If $\sum_{s=t_2-p_b+\Delta_2}^{t_2} x_{bs} = 0$ then the solution satisfies (10) because of (iv) and (vii). If $\sum_{s=t_2-p_b+\Delta_2}^{t_2} x_{bs} = 1$ then $\sum_{s=t_2-p_a+1}^{t_2+\Delta_2-1} x_{as} = 0$ because of (iii) and $\sum_{s=t_1-p_q+1}^{t_1+\Delta_1-1} x_{qs} = 0$ because of (vi) so the solution satisfies (10) because of (v).
Condition 6: If $\sum_{s=t_1-p_j+1}^{t_1-p_j+\Delta_1-1} x_{js} + \sum_{s=t_1-p_q+1}^{t_1-p_q+\Delta_1-1} x_{qs} = 0$ then the solution satisfies (10) because

of (i) and (iii). If $\sum_{s=t_1-p_j+1}^{t_1-p_j+\Delta_1-1} x_{js} + \sum_{s=t_1-p_q+1}^{t_1-p_q+\Delta_1-1} x_{qs} = 1$ then $\sum_{l\in T(j)\setminus\{q\}} \sum_{s=t_1-p_l+\Delta_1}^{t_1} x_{ls} = 0$ because of (ii) so the solution satisfies (10) because of (vi) and (vii).
Condition 7: If $\sum_{s=t_1+1}^{t_1+\Delta_1-1}(x_{js} + x_{qs}) = 0$ then the solution satisfies (10) because of (ii) and (iii). If $\sum_{s=t_1+1}^{t_1+\Delta_1-1}(x_{js} + x_{qs}) = 1$ then $\sum_{l\in T(j)\setminus\{q\}} \sum_{s=t_1-p_l+\Delta_1}^{t_1} x_{ls} = 0$ because of (i) so the solution satisfies (10) because of (vi) and (vii). □

6.3 Improving the LP relaxation

To see the effect of the valid inequalities, we have implemented a basic cutting-plane algorithm. Its working is illustrated in Figure 3. As the separation of the inequalities of van den Akker et al. (1999) takes too much computation time for the generation instances, we do not consider these inequalities in the computational experiments.

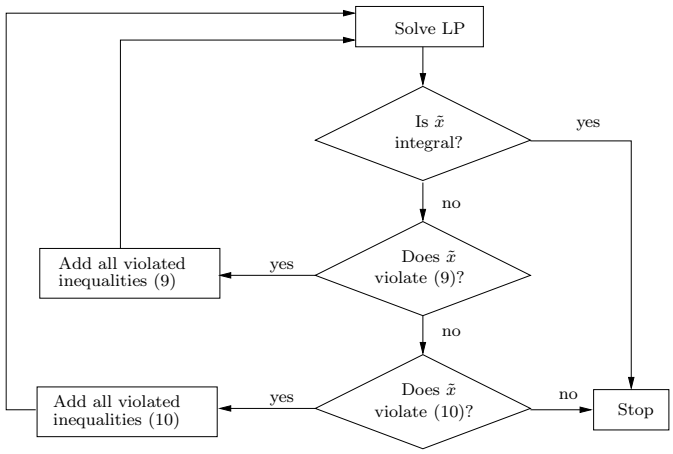


Figure 3: The basic cutting-plane algorithm

Table 2 provide statistics on this algorithm. Remark that preprocessing is not yet included in the algorithm here. We present the average total computation time as *time*, the average value of the LP solution as z_{LP} and the average value of the IP solution as z_{IP}. Recall that each row corresponds to 20 instances. Moreover, we provide statistics on the frequency with which optimal solutions are found. More precisely, we report the number of instances for which the optimal solution of the LP-relaxation is integral (see column n_{LP}), the number of instances for which the solution becomes integral after the addition of cuts (9) (in a cumulative way; see column $n_{(9)}$), and after addition of cuts (10) (in a cumulative way; see column $n_{(10)}$). For the instances whose LP-relaxation is fractional, the percentage of the gap ($z_{IP} - z_{LP}$) that is closed after adding valid inequalities is displayed. The percentage is computed as $100 \times \frac{z(\tilde{x})-z_{LP}}{z_{IP}-z_{LP}}$ where $z(\tilde{x})$ is the value found after adding the corresponding inequalities. When, in an extreme case, the LP solution and the IP solution have the same value, although the LP solution is fractional, we say that the gap is closed with 0% if the solution stays fractional after adding cuts and the gap is closed with 100% when the solution becomes integral.

We see that a small portion of the instances has an integral LP-solution, to be precise 5%. This percentage seems to grow mildly with the size of an instance. Adding inequalities (9)

Table 2: Impact of the valid inequalities (9) and (10)

$(m \times n)$	time	LP		LP + (9)		LP + (9),(10)		IP
		z_{LP}	n_{LP}	$GAP_{(9)}$	$n_{(9)}$	$GAP_{(10)}$	$n_{(10)}$	z_{IP}
1×100	2.35	774598.42	0	5.52%	0	not applicable		774884.65
1×150	8.67	1797688.70	0	4.07%	0	not applicable		1798358.50
1×200	19.61	3287610.99	0	2.55%	0	not applicable		3288583.75
2×100	107.15	232060.62	0	2.11%	0	2.60%	0	232110.20
2×150	329.87	575452.75	0	2.51%	0	2.67%	0	575646.10
2×200	613.17	1039188.77	0	1.22%	0	1.44%	0	1039433.30
3×100	62.30	77873.45	0	3.28%	0	3.58%	0	77892.05
3×150	174.64	197899.01	1	1.14%	1	1.85%	1	197972.50
3×200	396.12	388240.13	0	3.74%	0	3.95%	0	388400.85
5×100	14.54	6907.19	1	0.00%	1	18.82%	2	6909.00
5×150	72.29	21615.12	0	0.00%	0	2.94%	0	21620.50
5×200	200.40	47849.28	1	0.00%	1	1.18%	1	47861.15
10×100	10.52	2804.70	7	25.13%	9	48.03%	11	2805.25
10×150	74.69	10044.04	3	23.91%	5	31.45%	5	10045.20
10×200	83.21	22839.14	2	4.55%	2	12.45%	3	22839.90

helps in producing integral solutions: another 4 out of the 300 instances become integral and inequalities (10) yield another 4 instances. We conclude that both classes have a contribution in closing (part of) the gap. Inequalities (9) are quite powerful, bridging on average 5.32% of the gap. Inequalities (10) are also quite effective, bridging an additional 5.28% of the gap.

Next, we implement the same cutting-plane algorithm, but we now use variable fixing in each iteration. For a first set of statistics, we use the objective value for the MIP found by CPLEX within one hour of computation time as an upper bound. For a second set of statistics, the upper bound is obtained by a heuristic procedure. To obtain an upper bound on the optimal solution, we run CPLEX for one minute. If at least one feasible solution with a GAP smaller than 3% is found, the objective value of the best solution is fixed as an upper bound. If no such feasible solution is found, we rerun CPLEX until one is found. Remark that CPLEX produces the Driebeek penalties as part of the output.

Table 3 provide statistics on the first implementation. This can be seen as the ideal scenario for the preprocessing technique as the upper bound that is used is the optimal solution for the integer problem (or the best value obtained by CPLEX in one hour of computation time). We see that, although using preprocessing without valid inequalities hardly succeeds in closing the gap, adding preprocessing on top of the valid inequalities helps further in producing integral solutions and in closing the gap. Adding inequalities (9) makes that another 11 out of the 300 instances become integral and bridge on average 7.99% of the gap. Inequalities (10) yield another 7 integral instances and bridge on average 4.88% of the gap. Moreover, the average computation times are smaller compared to the implementation without preprocessing. Table 4 provide statistics on the second implementation. We see that adding preprocessing helps further in producing integral solutions and in closing the gap.

Table 3: Impact of preprocessing with the objective value for the MIP found by CPLEX as an upper bound

$(m \times n)$	time	LP		LP + pre + (9)		LP + pre + (9),(10)		IP
		z_{LP}	n_{LP}	$GAP_{(9)}$	$n_{(9)}$	$GAP_{(10)}$	$n_{(10)}$	z_{IP}
1×100	8.84	774598.42	0	5.65%	0	not applicable		774884.65
1×150	23.45	1797688.70	0	4.07%	0	not applicable		1798358.50
1×200	64.05	3287610.99	0	2.55%	0	not applicable		3288583.75
2×100	78.53	232060.62	0	2.11%	0	2.82%	0	232110.20
2×150	220.84	575452.75	0	2.51%	0	2.67%	0	575646.10
2×200	385.63	1039188.77	0	1.22%	0	1.44%	0	1039433.20
3×100	48.20	77873.45	0	3.34%	0	3.64%	0	77892.05
3×150	159.19	197899.01	1	1.15%	1	1.85%	1	197972.50
3×200	353.05	388240.13	0	4.01%	0	4.22%	0	388400.85
5×100	12.62	6907.19	1	7.02%	2	23.64%	5	6909.00
5×150	69.88	21615.12	0	5.00%	1	7.94%	1	21620.50
5×200	159.35	47849.28	1	0.01%	1	1.18%	1	47861.15
10×100	8.76	2804.70	7	41.67%	12	58.79%	14	2805.25
10×150	36.05	10044.04	3	23.91%	5	33.66%	6	10045.20
10×200	45.80	22839.14	2	15.66%	4	24.37%	5	22839.90

Table 4: Impact of preprocessing with the objective value for the heuristic procedure as an upper bound

$(m \times n)$	LP		LP + pre + (9)		LP + pre + (9),(10)		IP
	z_{LP}	n_{LP}	$GAP_{(9)}$	$n_{(9)}$	$GAP_{(10)}$	$n_{(10)}$	z_{IP}
1×100	774598.42	0	5.65%	0	not applicable		774884.65
1×150	1797688.70	0	4.07%	0	not applicable		1798358.50
1×200	3287610.99	0	2.55%	0	not applicable		3288583.75
2×100	232060.62	0	2.11%	0	2.82%	0	232110.20
2×150	575452.75	0	2.51%	0	2.67%	0	575646.10
2×200	1039188.77	0	1.22%	0	1.44%	0	1039433.30
3×100	77873.45	0	3.34%	0	3.64%	0	77892.05
3×150	197899.01	1	1.15%	1	1.85%	1	197972.50
3×200	388240.13	0	3.74%	0	3.95%	0	388400.85
5×100	6907.19	1	7.02%	2	23.64%	5	6909.00
5×150	21615.12	0	5.00%	1	7.94%	1	21620.50
5×200	47849.28	1	0.01%	1	1.18%	1	47861.15
10×100	2804.70	7	38.46%	12	58.79%	14	2805.25
10×150	10044.04	3	23.91%	5	33.66%	6	10045.20
10×200	22839.14	2	15.66%	4	24.37%	5	22839.90

Adding inequalities (9) makes that another 11 out of the 300 instances become integral and bridge on average 7.76% of the gap. Inequalities (10) yield another 7 integral instances and bridge on average 5.15% of the gap. We can conclude that the solution of our heuristic is a good alternative for the objective value of CPLEX as an upper bound for preprocessing.

6.4 Exact solution of the problem

In this section we focus on the exact solution of the IP formulation. A first approach consists in directly solving the IP formulation by CPLEX with a time limit of one hour (referred to as $InitIP$). The second approach (referred to as $Alg2(0)$) consists in applying the preprocessing technique inside a cutting-plane algorithm, as described in Algorithm 2, before solving by CPLEX the modified instance. The latter may have fixed variables x_{jt} and added valid inequalities. To evaluate the potential efficiency of the preprocessing technique, we give the solution of the first approach as an upper bound. Therefore, as soon as $InitIP$ finds the optimal solution, we use in $Alg2(0)$ the best possible upper bound leading to fix a large number of fixed variables. However, to evaluate the practical efficiency of our algorithm, we implemented a third approach (referred to as $Alg2(1)$) for which the upper bound for the preprocessing is now obtained by a heuristic procedure ($UB = z^*_{Heur}$). The processing time of the heuristic is included in the total computation time of the algorithm. For experimentation, we only consider instances with one, two or three machines since the generated instances start to be easily solvable (in a few seconds) by CPLEX as the number of machines increases.

Algorithm 2 Exact solution with an initial preprocessing; input: UB

Run Algorithm 1 with UB and no valid inequalities: x^*_{LP} denotes the returned variable values.
if (x^*_{LP} is not integral) **then**
  Iterate=$true$.
end if
while (Iterate=$true$) **do**
  Starting with x^*_{LP}, let VI be the set of violated inequalities (9).
  if ($|VI| \neq \emptyset$) **then**
    Run Algorithm 1 with UB and VI: x^*_{LP} denotes the returned variable values.
  else
    Iterate=$false$.
  end if
end while
Convert the LP with added valid inequalities and fixed variables into an IP model.
Solve the resulting IP: z^*_{IP} is the computed optimal solution value.
return z^*_{IP}.

In Table 5, we only look at instances that were solved to optimality by all three implementations. The number of such instances (out of twenty) is given in the second column. The average computation time and the average number of nodes that CPLEX explores, are displayed in columns *time* and *#nodes* for all algorithms. For $Alg2(0)$ and $Alg2(1)$

we also provide in column $fixed$ the average percentage of variables that are fixed by the preprocessing.

Looking at the results of $InitIP$, we notice that the instances seem to be easier for CPLEX as the number of machines increase. Remark that the cutting-plane algorithm with preprocessing may penalize the solution by CPLEX: effort is spent in generating valid inequalities and fixing variables in $Alg2(0)$ and $Alg2(1)$ while $InitIP$ is efficient. To illustrate that, look at the instances with 3 machines and 100 jobs. On average, these instances are solved by CPLEX in only 14.36 seconds, so we can not expect a large gain in solution time. Indeed, generating valid inequalities and fixing variables lead to an increased overall CPU time (97.36 seconds for $Alg2(0)$) even if the number of explored nodes is reduced and more than 99% of the variables are fixed by preprocessing. Therefore, the conclusion seems to be that $Alg2(0)$ is more suited for really "hard" instances. The efficiency of the cutting-plane algorithm with preprocessing decreases as the upper bound becomes weaker, since the results of $Alg2(1)$ are generally speaking worse than those of $Alg2(0)$. For that reason, table 5 highlights an interesting possible future research: improving the quality of the heuristic algorithm to obtain better final solutions for the problem.

Table 6 contains the instances for which at least one of the three implementations was not able to solve to optimality due to the time and memory limits imposed (see section 6.1.1). For each implementation, we mention the average computation time. Moreover, the number of instances out of the considered subset that are solved to optimality and the average GAP is displayed. The GAP is computed as follows for a given algorithm E: $\text{GAP}(E) = 1000000 * \frac{z_E - z^{min}}{z_E}$, with z_E the objective value returned by algorithm E and $z^{min} = \min(z_{InitIP}; z_{Alg2(0)}; z_{Alg2(1)})$. Finally, the average number of fixed variables is given for Alg2(0) and Alg2(1). Remark that these solutions are not always guaranteed to be optimal. We notice that the GAP is usually very small for the algorithm $Alg2(1)$, especially for the one machine instances. This conclusion is in line with the result in Table 5. As soon as instances are hard to solve, the cutting-plane algorithm with preprocessing may enable to find a better solution than $InitIP$ in 3600 seconds.

In general, we can state that $InitIP$ performs rather well and that the cutting-plane algorithm with preprocessing does not really outperforms it. However, as soon as we have to solve hard instances, it turns out that the the cutting-plane algorithm with preprocessing yields better results. Notice that we have not been able to do preprocessing at each node of the branch-and-cut algorithm of CPLEX due to a lack of interactions with the mathematical solver. This would drastically improve the results of $Alg2(0)$ and $Alg2(1)$.

7 Conclusion

We modeled the unrelated parallel machine scheduling problem where the processing cost of each job is an arbitrary function of its starting time as a single machine scheduling problem using a time-indexed formulation. We have shown that valid inequalities from literature for single-machine problems can be applied to multi-machine problems. A new set of facet-inducing inequalities has been presented, and a cutting-plane algorithm has been proposed. We have also implemented a preprocessing technique within that algorithm to try to reduce the size of the instances to solve. Computational experiments have been conducted and they

Table 5: Analysis of instances that were solved to optimality by all implementations

		InitIP		Alg2(0)			Alg2(1)		
$(m \times n)$	#	time	# nodes	time	# nodes	fixed	time	# nodes	fixed
1×100	19	335.06	2426.16	269.30	2320.58	70.36%	375.73	1997.47	54.11%
1×150	9	1237.32	3925.67	943.14	3963.33	69.95%	1681.32	5104.33	24.07%
2×100	20	35.48	350.1	168.52	378.40	96.11%	194.86	291.20	95.86%
2×150	17	672.27	4149.76	786.83	3708.94	84.94%	1197.71	4647.94	30.66%
2×200	9	1459.26	4128.44	1278.33	3215.00	90.44%	1712.49	4342.89	34.04%
3×100	20	14.36	62.00	97.36	48.15	99.03%	54.62	48.15	99.03%
3×150	20	96.81	1183.70	341.96	1271.15	97.47%	199.58	702.80	94.27%
3×200	19	530.10	2113.26	898.96	1906.89	94.84%	739.93	2196.89	67.07%

Table 6: Analysis of instances that were not solved to optimality

		InitIP			Alg2(0)				Alg2(1)			
$(m \times n)$	#	time	# opt	GAP	time	# opt	GAP	fixed	time	# opt	GAP	fixed
1×100	1	3600.00	0	59.83%	3600.00	0	26.18%	42.98%	3600.00	0	0.00%	34.63%
1×150	11	2799.81	5	24.15%	2995.97	3	8.85%	44.63%	3302.85	2	7.50%	25.00%
1×200	20	3600.00	0	33.25%	3600.00	0	36.54%	42.07%	3600.00	0	30.32%	17.88%
2×150	3	3176.04	2	0.00%	3563.02	2	15.08%	74.52%	3067.32	1	0.00%	6.51%
2×200	11	3600.00	3	17.07%	3479.78	5	20.44%	78.10%	3552.09	1	50.33%	26.47%
3×200	1	1348.20	1	0.00%	2573.34	1	0.00%	87.75%	3600.00	0	15.60%	37.22%

show that the valid inequalities lead to strengthen the linear relaxation of the time-indexed formulation. These experiments also show that the proposed cutting-plane algorithm with preprocessing may help to solve the instances which are hard to solve for the mathematical solver more efficiently.

As future research directions, it would be interesting to implement a faster separation algorithm for the inequalities of van den Akker et al. (1999) to improve the exact solution of the problem by the cutting-plane algorithm with preprocessing. It would also be interesting to provide a very fast and good heuristic algorithm to get a good preprocessing. At last, the possibly most promising research line from an experimental point of view is certainly to integrate the preprocessing at each node of the branch-and-cut algorithm used to solve the time-indexed formulation.

8 Appendix

Theorem *Inequalities (10) are facet-defining inequalities for P_m, for each $m \geq 2$.*

Proof: Consider formulation (5)-(8) that we restate here: For each job i and machine k, and for each time period $t = 1, \ldots, T - p_{ik} + 1$, we define the decision variables

$$x_{i,k,t} = \begin{cases} 1 & \text{if job } i \text{ starts on machine } k \text{ at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

An IP-model for this machine scheduling problem is the following:

$$\min \sum_{i=1}^{n} \sum_{k=1}^{m} \sum_{t=1}^{T-p_{ik}+1} c_{i,k,t} x_{i,k,t} \tag{13}$$

subject to

$$\sum_{k=1}^{m} \sum_{t=1}^{T-p_{ik}+1} x_{i,k,t} = 1 \qquad \forall i = 1, \ldots, n, \tag{14}$$

$$\sum_{i=1}^{n} \sum_{s=t-p_{ik}+1}^{t} x_{i,k,s} \leq 1 \qquad \forall k = 1, \ldots, m; \forall t = \min_i p_{ik}, \ldots, T, \tag{15}$$

$$x_{i,k,t} \in \{0, 1\} \quad \forall i = 1, \ldots, n; \forall k = 1, \ldots, m; \forall t = 1, \ldots, T - p_{ik} + 1. \tag{16}$$

Observe that this formulation is *identical* to formulation (5)-(8); we only use different indices here (see the paragraph preceding (5)-(8)).

Let us first establish the dimension of the corresponding polytope P_m.

Lemma 5 *$dim(P_m) = n(mT + m - 1) - \sum_{i=1}^{n} \sum_{k=1}^{m} p_{ik}$.*

Proof: How many variables exist in (5)-(8)? Consider some job i, $1 \leq i \leq n$. In case machine k ($1 \leq k \leq m$), performs this job, there are $T - p_{ik} + 1$ possible moments to start this job, giving rise to equally many binary variables. Hence the total number of variables in (5)-(8)

equals: $\sum_{i=1}^{n}\sum_{k=1}^{m}(T-p_{ik}+1) = n(mT+m) - \sum_{i=1}^{n}\sum_{k=1}^{m} p_{ik}$. Since we have n linearly independent inequalities (14) it easily follows that $\dim(P_m) \leq n(mT+m-1)-\sum_{i=1}^{n}\sum_{k=1}^{m} p_{ik}$. We will now show that, in fact, equality holds.

Consider some equality that is valid for all feasible solutions x:

$$\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1} \pi_{i,k,t}x_{i,k,t} = \pi_0. \tag{17}$$

We now identify a particular feasible solution, denoted by solution S, as follows. Let some job i ($1 \leq i \leq n$) start on machine k ($1 \leq k \leq m$) at time s ($1 \leq s \leq T - p_{ik} + 1$); all other jobs start on machine ℓ, $\ell \neq k$, in a way that no job on machine ℓ overlaps the moments $\{s, s+1, \ldots, s+p_{i,\ell}-1\}$ (notice that this is always possible, if T is large enough, sat $T \geq 2 \times \max_k \sum_i p_{ik}$).

Consider now a feasible solution that is identical to S except that job i starts on machine k at time t, $t \neq s$. Since equality (17) holds for all feasible solutions, it follows that

$$\pi_{i,k,s} = \pi_{i,k,t} \qquad \forall i = 1,\ldots,n; \forall k = 1,\ldots,m; \forall s,t \in \{1,\ldots,T-p_{ik}+1\}. \tag{18}$$

Consider now another feasible solution that is identical to S except that job i starts also on machine ℓ at time s. Again, since equality (17) holds for all feasible solutions, it follows that

$$\pi_{i,k,s} = \pi_{i,\ell,s} \qquad \forall i = 1,\ldots,n; \forall k,\ell \in \{1,\ldots,m\}; \forall s = 1,\ldots,T-p_{ik}+1. \tag{19}$$

Using (18) and (19), we conclude that there exist multipliers π_i such that

$$\pi_i = \pi_{i,k,t} \qquad \forall i = 1,\ldots,n; \forall k = 1,\ldots,m; \forall t = 1,\ldots,T-p_{ik}+1.$$

Thus, we can rewrite equality (17) as follows:

$$\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1} \pi_{i,k,t}x_{i,k,t} = \sum_{i=1}^{n}\pi_i\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1} x_{i,k,t} = \pi_0,$$

thereby showing that equality (17) is a linear combination of equalities (14). This proves the lemma. □

We now proceed to prove that inequalities (10) are facet-defining. In order to facilitate the description of the proof, we define the following intervals of time-units. Given jobs i_1, i_2, machines k_1, k_2, time-units t_1, t_2, and parameters Δ_1, Δ_2, we define:

$$\begin{aligned} A &= [t_1 - p_{i_1,k_1} + 1, \ldots, t_1 + \Delta_1 - 1], \\ B &= [t_1 - p_{i_2,k_1} + 1, \ldots, t_1 + \Delta_1 - 1], \\ C_i &= [t_1 - p_{i,k_1} + \Delta_1, \ldots, t_1](i \neq i_1, i \neq i_2), \\ D &= [t_2 - p_{i_1,k_2} + 1, \ldots, t_2 + \Delta_2 - 1], \\ E &= [t_2 - p_{i_2,k_2} + \Delta_2, \ldots, t_2]. \end{aligned}$$

This allows us to rewrite inequality (10) as follows:

$$\sum_{s\in A} x_{i_1,k_1,s} + \sum_{s\in B} x_{i_2,k_1,s} + \sum_{i:i\neq i_1, i\neq i_2}\sum_{s\in C_i} x_{i,k_1,s} + \sum_{s\in D} x_{i_1,k_2,s} + \sum_{s\in E} x_{i_2,k_2,s} \leq 2. \tag{20}$$

Let $F = \{x \in P_m$: x satisfies (20) with equality$\}$, and consider some equality that is valid for all $x \in F$:

$$\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1} \pi_{i,k,t}x_{i,k,t} = \pi_0.$$

We will show that there exist multipliers ρ_i $(1 \leq i \leq n)$, and α such that:

$$\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1} \pi_{i,k,t}x_{i,k,t} = \sum_{i=1}^{n}\rho_i\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1} x_{i,k,t} +$$

$$\alpha\left(\sum_{s\in A} x_{i_1,k_1,s} + \sum_{s\in B} x_{i_2,k_1,s} + \sum_{i:i\neq i_1,i\neq i_2}\sum_{s\in C_i} x_{i,k_1,s} + \sum_{s\in D} x_{i_1,k_2,s} + \sum_{s\in E} x_{i_2,k_2,s}\right) = \pi_0.$$

This shows that any equality valid for all feasible solutions, is a linear combination of the equalities (14) and the equality corresponding to (20). In our proof we will use feasible solutions in F, and sometimes use the phrase "all other jobs start in some feasible way on some machine k". Our assumption that T is large enough guarantees that there is indeed some way to place those jobs on that machine.

Consider now a solution called S_1, where job i_1 starts on machine k_1 at time $t_1 - p_{i_1,k_1} + 1$, where job i_2 starts on machine k_1 at time $t_1 + 1$, where some job i, $(i \neq i_1, i \neq i_2)$ starts on some machine k $(k \neq k_1)$ at time s, with $1 \leq s \leq T - p_{ik} + 1$, and where all other jobs start, in some feasible way, on machine k_1. We modify solution S_1 by keeping everything the same except that we start job i on machine k at time t, $t \neq s$. Clearly, both solution S_1, and the modified solution are in F. It follows that

$$\pi_{i,k,s} = \pi_{i,k,t} \quad \forall i \in \{1,\ldots,n\} \setminus \{i_1,i_2\}; \forall k = 1,\ldots,m; k \neq k_1; \forall s,t \in \{1,\ldots,T-p_{ik}+1\}. \tag{21}$$

Also, we can modify S_1 by shifting job i to machine ℓ, $\ell \neq k, \ell \neq k_1$ at time s, to arrive at:

$$\pi_{i,k,s} = \pi_{i,\ell,s} \quad \forall i \in \{1,\ldots,n\}\setminus\{i_1,i_2\}; \forall k,\ell \in \{1,\ldots,m\}\setminus\{k_1\}; \forall s = 1,\ldots,T-p_{ik}+1. \tag{22}$$

Together, equalities (21) and (22) imply:

$$\pi_{i,k,t} = \rho_i \qquad \forall i \in \{1,\ldots,n\} \setminus \{i_1,i_2\}; \forall k = 1,\ldots,m; k \neq k_1; \forall t = 1,\ldots,T-p_{ik}+1. \tag{23}$$

We will now proceed to argue that (23) is in fact also valid for machine k_1 when $t \notin C_i$, $1 \leq i \leq n$, i.e., we will show that:

$$\pi_{i,k_1,t} = \rho_i \qquad \forall i \in \{1,\ldots,n\} \setminus \{i_1,i_2\}; \forall t \notin C_i. \tag{24}$$

We assume here, for reasons of convenience (and without loss of generality) that t_1 is such that there is ample space both to the left of t_1, as well as to the right of t_1. More precisely, we assume that $p_{i,k_1} \leq t_1 \leq T - p_{i,k_1}$ for all i. Consider now solution S_2, where job i_1 starts on machine k_1 at time $t_1 - p_{i_1,k_1} + 1$, where job i_2 starts on machine k_2 at time $t_2 - p_{i_2,k_2} + \Delta_2$, where some job i starts on machine k_1 at time s, $s \geq t_1 + 1$, and where all other jobs are placed in some feasible way on machine k_2 leaving free the time units $[s,\ldots,s+p_{ik_2}-1]$. We

modify solution S_2 by keeping everything the same except that we start job i on machine k_1 at time $t = 1$. Clearly, both solution S_2, and the modified solution are in F. It follows that:

$$\pi_{i,k_1,s} = \pi_{i,k_1,1} \qquad \forall i \in \{1,\ldots,n\} \setminus \{i_1,i_2\}; \forall s = t_1+1,\ldots,T-p_{i,k_1}+1. \tag{25}$$

We can also modify S_2 by keeping everything the same except that we start job i on machine some k ($k \neq k_1$) at time s, implying (using (23)):

$$\pi_{i,k_1,s} = \pi_{i,k,s} = \rho_i \quad \forall i \in \{1,\ldots,n\} \setminus \{i_1,i_2\}; \forall k = 1,\ldots,m; k \neq k_1; \forall s = 1,\ldots,T-p_{i,k_1}+1. \tag{26}$$

Consider now solution S_3, where job i_1 starts on machine k_1 at time $t_1 + \Delta_1 - 1$, where job i_2 starts on machine k_2 at time $t_2 - p_{i_2,k_2} + \Delta_2$, where some job i starts on machine k_1 at time s, $s \leq t_1 - p_{i,k_1} + \Delta_1 - 1$, and where all other jobs are placed in some feasible way on machine k_2. We modify solution S_3 by keeping everything the same except that we start job i on machine k_1 at time $t = 1$. Clearly, both solution S_3, and the modified solution are in F. It follows that:

$$\pi_{i,k_1,s} = \pi_{i,k_1,1} \qquad \forall i \in \{1,\ldots,n\} \setminus \{i_1,i_2\}; \forall s = 1,\ldots,t_1-p_{i,k_1}+\Delta_1-1. \tag{27}$$

Together, the conditions (25), (26) and (27) imply (24).

Consider now solution S_4 where job i_1 starts on machine k_2 at time $t_2 - p_{i_1,k_2} + 1$, where job i, $i \neq i_1, i \neq i_2$ starts on machine k_1 at time $t_1 - p_{i,k_1} + \Delta_1$, where job i_2 starts on some machine k, $k \neq k_1, k \neq k_2$ at some time s, and where all other jobs are placed in a feasible way on machine k_1. We modify solution S_4 by keeping everything the same except that we place job i_2 on machine k at time t ($t \neq s$). Clearly, both solution S_4, and the modified solution are in F. We get:

$$\pi_{i_2,k,s} = \pi_{i_2,k,t} \qquad \forall k \in \{1,\ldots,m\} \setminus \{k_1,k_2\}; \forall s,t \in \{1,\ldots,T-p_{i_2,k}+1\}. \tag{28}$$

We can also modify S_4 by keeping everything the same except that we shift job i_2 to some other machine ℓ ($\ell \neq k_1, \ell \neq k_2$) at time s, implying:

$$\pi_{i_2,k,s} = \pi_{i_2,\ell,s} \qquad \forall k,\ell \in \{1,\ldots,m\} \setminus \{k_1,k_2\}; \forall s = 1,\ldots,T-p_{i_2,k}+1. \tag{29}$$

Together, the conditions (28), and (29) imply:

$$\pi_{i_2,k,t} = \rho_{i_2} \qquad \forall k \in \{1,\ldots,m\} \setminus \{k_1,k_2\}; \forall t = 1,\ldots,T-p_{i_2,k}+1. \tag{30}$$

A similar construction can be used to infer:

$$\pi_{i_1,k,t} = \rho_{i_1} \qquad \forall k \in \{1,\ldots,m\} \setminus \{k_1,k_2\}; \forall t = 1,\ldots,T-p_{i_1,k}+1. \tag{31}$$

Consider solution S_5 where job i_1 starts on machine k_1 at time s, $s \notin A$, where job i_2 starts on machine k_2 at time t_2, where job i starts on k_1 at time $t_1 - p_{i,k_1} + \Delta_1$, and where all other jobs are placed in a feasible way on machine k_2. We modify solution S_5 by keeping everything the same except that we place job i_1 on machine k_1 at time t ($t \notin A$). Clearly, both solution S_5, and the modified solution are in F. Using constructions like these, we get:

$$\pi_{i_1,k_1,s} = \pi_{i_1,k_1,t} \qquad \forall s,t \notin A. \tag{32}$$

We can also modify S_5 by keeping everything the same except that we shift job i_1 to some other machine k ($k \neq k_1$) at time s, implying:

$$\pi_{i_1,k_1,s} = \pi_{i_1,k,s} \qquad \forall k = 1,\ldots,m; k \neq k_1; \forall s = 1,\ldots,T - p_{i_1,k_1} + 1. \tag{33}$$

Together, the conditions (32), and (33) imply:

$$\pi_{i_1,k_1,t} = \rho_{i_1} \qquad \forall t \notin A. \tag{34}$$

Similar constructions can be used to infer:

$$\pi_{i_1,k_2,t} = \rho_{i_1} \qquad \forall t \notin D, \tag{35}$$

$$\pi_{i_2,k_1,t} = \rho_{i_2} \qquad \forall t \notin B, \tag{36}$$

and

$$\pi_{i_2,k_2,t} = \rho_{i_2} \qquad \forall t \notin E. \tag{37}$$

Consider solution S_6 where job i_1 starts on machine k_1 at time s, $s \in A$, where job i_2 starts on machine k_2 at time $t_2 - p_{i_2,k_2} + \Delta_2$, and where all other jobs are placed in a feasible way on machine k_2. We modify solution S_6 by keeping everything the same except that we place job i_1 on machine k_1 at time t ($t \neq s, t \in A$). Clearly, both solution S_6, and the modified solution are in F. We get:

$$\pi_{i_1,k_1,s} = \pi_{i_1,k_1,t} \qquad \forall s,t \in A. \tag{38}$$

In a similar fashion, we can derive:

$$\pi_{i_1,k_2,s} = \pi_{i_1,k_2,t} \qquad \forall s,t \in D. \tag{39}$$

Moreover, consider solution S_7 where job i_1 starts on machine k_1 at time $t_1 - p_{i_1,k_1} + 1$, where job i_2 starts on machine k_1 at time $t_1 + \Delta_1 - 1$, and where all other jobs are placed in a feasible way on machine k_2, leaving free the time units $[t_2 - p_{i_1,k_2} + 1,\ldots,t_2]$. We modify solution S_7 by keeping everything the same except that we place job i_1 on machine k_2 at time $t_2 - p_{i_1,k_2} + 1$. Clearly, both solution S_7, and the modified solution are in F. We get:

$$\pi_{i_1,k_1,t_1-p_{i_1,k_1}+1} = \pi_{i_1,k_2,t_2-p_{i_1,k_2}+1}. \tag{40}$$

Using conditions (38), (39) and (40), we find:

$$\pi_{i_1,k_1,t} = \pi_{i_1,k_2,s} \equiv \pi^{in}_{i_1} \qquad \forall t \in A; \forall s \in D. \tag{41}$$

In a similar fashion, we can derive:

$$\pi_{i_2,k_1,t} = \pi_{i_2,k_2,s} \equiv \pi^{in}_{i_2} \qquad \forall t \in B; \forall s \in E, \tag{42}$$

and:

$$\pi_{i,k_1,t} = \pi_i^{in} \qquad \forall i \in \{1,\ldots,n\} \setminus \{i_1,i_2\}; \forall t \in C_i. \tag{43}$$

Consider solution S_8 where job i_1 starts on machine k_2 at time $t_2 - p_{i_1,k_2} + 1$, where job i_3 ($i_3 \neq i_2$) starts on machine k_1 at time t_1 (notice that $t_1 \in C_{i_3}$), where job i_4 ($i_4 \neq i_2, i_4 \neq i_3$) starts on machine k_1 at time s with $s \geq t_1 + p_{i_3,k_1} + p_{i_4,k_1}$, and where all other jobs are somewhere on machine k_2. We modify solution S_8 by keeping everything the same except that we interchange jobs i_3 and i_4, ie, by starting job i_3 on machine k_1 at time s, and job i_4 on machine k_1 at time t_1. Clearly, both solution S_8, and the modified solution are in F. Then, we find that:

$$\pi_{i_3,k_1,t_1} + \pi_{i_4,k_1,s} = \pi_{i_3,k_1,s} + \pi_{i_4,k_1,t_1}.$$

Using (24) and (43), this is equivalent to:

$$\pi_{i_3,k_1,t_1} - \rho_{i_3} = \pi_{i_4,k_1,t_1} - \rho_{i_4} \equiv \alpha,$$

or, by extending the construction for any time $t \in C_i$

$$\pi_{i,k_1,t} - \rho_i = \alpha \qquad \forall i \in \{1,\ldots,n\} \setminus \{i_1,i_2\}; \forall t \in C_i. \tag{44}$$

Consider a solution S_9 where job i_1 starts on machine k_1 at time t_1, where job i_2 starts on machine k_2 at time t_2 (notice that $t_2 \in E$), where some job i ($i \neq i_1, i \neq i_2$) starts on machine k_1 at time s with $s \geq t_1 + p_{i_1,k_1} + p_{i,k_1}$, and where all other jobs are somewhere on machine k_2. We modify solution S_9 by keeping everything the same except that we interchange jobs i_1 and i, i.e., by starting job i on machine k_1 at time t_1, and job i_1 on machine k_1 at time s. Clearly, both solution S_9, and the modified solution are in F. Then, we find that:

$$\pi_{i_1,k_1,t_1} + \pi_{i,k_1,s} = \pi_{i_1,k_1,s} + \pi_{i,k_1,t_1}.$$

Using (24), (34) and (41), this is equivalent to:

$$\pi_{i_1,k_1,t_1} - \rho_{i_1} = \pi_{i,k_1,t_1} - \rho_i = \alpha,$$

or, in fact:

$$\pi_{i_1,k_1,t} - \rho_{i_1} = \pi_{i_1,k_2,s} - \rho_{i_1} = \alpha \qquad \forall t \in A; \forall s \in D. \tag{45}$$

A similar construction using (42) allows us to conclude:

$$\pi_{i_2,k_1,t} - \rho_{i_2} = \pi_{i_2,k_2,s} - \rho_{i_2} = \alpha \qquad \forall t \in B; \forall s \in E. \tag{46}$$

We are now finally able to derive the result:

$$\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1}\pi_{i,k,t}x_{i,k,t} =$$

$$\sum_{i:i\neq i_1,i\neq i_2}\sum_{k\neq k_1}(\sum_{t=1}^{T-p_{ik}+1}\pi_{i,k,t}x_{i,k,t}) + \sum_{t\notin C_i}\pi_{i,k_1,t}x_{i,k_1,t} + \sum_{t\in C_i}\pi_{i,k_1,t}x_{i,k_1,t} +$$

$$\sum_{k\neq k_1,k\neq k_2}(\sum_{t=1}^{T-p_{i_1,k}+1}\pi_{i_1,k,t}x_{i_1,k,t} + \sum_{t=1}^{T-p_{i_2,k}+1}\pi_{i_2,k,t}x_{i_2,k,t})$$

$$\sum_{t\in A}\pi_{i_1,k_1,t}x_{i_1,k_1,t} + \sum_{t\in D}\pi_{i_1,k_2,t}x_{i_1,k_2,t} + \sum_{t\notin A}\pi_{i_1,k_1,t}x_{i_1,k_1,t} + \sum_{t\notin D}\pi_{i_1,k_2,t}x_{i_1,k_2,t} +$$

$$\sum_{t\in B}\pi_{i_2,k_1,t}x_{i_2,k_1,t} + \sum_{t\in E}\pi_{i_2,k_2,t}x_{i_2,k_2,t} + \sum_{t\notin B}\pi_{i_2,k_1,t}x_{i_2,k_1,t} + \sum_{t\notin E}\pi_{i_2,k_2,t}x_{i_2,k_2,t}.$$

By plugging in the values we found for the $\pi_{i,k,t}$ coefficients derived in the conditions respectively (23), (24), (44), (31), (30), (45), (34), (35), (46), (36) and (37), we find:

$$\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1}\pi_{i,k,t}x_{i,k,t} = \sum_{i=1}^{n}\rho_i\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1}x_{i,k,t} +$$

$$\alpha(\sum_{s\in A}x_{i_1,k_1,s} + \sum_{s\in B}x_{i_2,k_1,s} + \sum_{i:i\neq i_1,i\neq i_2}\sum_{s\in C_i}x_{i,k_1,s} + \sum_{s\in D}x_{i_1,k_2,s} + \sum_{s\in E}x_{i_2,k_2,s}) = \pi_0, \qquad \square$$

thereby proving our result.

References

P. Baptiste, F. Della Croce, A. Grosso, and V. T'Kindt. Sequencing a single machine with due dates and deadlines: an ILP-based approach to solve very large instances. *Journal of Scheduling*, 13(1):39–47, 2010.

L. Bigras, M. Gamache, and G. Savard. Time-indexed formulations and the total weighted tardiness problem. *INFORMS Journal on Computing*, 20(1):133–142, 2008.

Y. Crama and F. C. R. Spieksma. Scheduling jobs of equal length: Complexity, facets and computational results. *Mathematical Programming*, 72:207–227, 1996.

Y. Crama, A. W. J. Kolen, A. G. Oerlemans, and F. C. R. Spieksma. Throughput rate optimization in the automated assembly of printed circuit boards. *Annals of Operations Research*, 26:455–480, 1990.

M. E. Dyer and L. A. Wolsey. Formulating the single machine sequencing problem with release dates as a mixed integer problem. *Discrete Applied Mathematics*, 26:255–270, 1990.

M. Gairing, B. Monien, and A. Woclaw. A faster combinatorial approximation algorithm for scheduling unrelated parallel machines. *Theoretical Computer Science*, 380:87–99, 2007.

P. Hansen, C. Oğuz, and N. Mladenović. Variable neighborhood search for minimum cost berth allocation. *European Journal of Operational Research*, 191(3):636–649, 2008.

J. K. Lenstra, D. B. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

F. Sourd. New exact algorithms for one-machine earliness-tardiness scheduling. *INFORMS Jounal of Computing*, 21:167–175, 2009.

J. P. Sousa and L. A. Wolsey. A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical Programming*, 54:353–367, 1992.

S. Tanaka, S. Fujikuma, and M. Araki. An exact algorithm for single-machine scheduling without machine idle time. *Journal of Scheduling*, 12:575–593, 2009.

V. T'kindt, F. Della Croce, and J. L. Bouquard. Enumeration of pareto optima for a flowhop scheduling problem with two criteria. *INFORMS Journal on Computing*, 19(1): 64–72, 2007.

J.A. Tomlin. An improved branch-and-bound method for integer programming. *Operations Research*, 19(4):1070–1075, 1971.

Y. Unlu and S. J. Mason. Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers & Industrial Engineering*, 58:785–800, 2010.

J. M. van den Akker, J. A. Hoogeveen, and S. L. van de Velde. Parallel machine scheduling by column generation. *Operations Research*, 47(6):862–872, 1999.

J. M. van den Akker, C. A. J. Hurkens, and M. W. P. Savelsbergh. Time-indexed formulations for machine scheduling problems: Column generation. *INFORMS Journal on Computing*, 12(2):111–124, 2000.

J.M. van den Akker, C.P.M. Van Hoesel, and M.W.P. Savelsbergh. A polyhedral approach to single-machine scheduling problems. *Mathematical Programming*, 85:541–572, 1999.

T. Vredeveld and C. Hurkens. Experimental comparison of approximation algorithms for scheduling unrelated parallel machines. *INFORMS Journal on Computing*, 14(2):175–189, 2002.

FACULTY OF ECONOMICS AND BUSINESS
Naamsestraat 69 bus 3500
3000 LEUVEN, BELGIË
tel. + 32 16 32 66 12
fax + 32 16 32 67 91
info@econ.kuleuven.be
www.econ.kuleuven.be